On the Application of Active Learning to Handle Data Evolution in Android Malware Detection

Alejandro Guerra-Manzanares(✉) and Hayretdin Bahsi

Tallinn University of Technology, Tallinn, Estonia
{alejandro.guerra,hayretdin.bahsi}@taltech.ee

Abstract. Mobile malware detection remains a significant challenge in the rapidly evolving cyber threat landscape. Although the research about the application of machine learning methods to this problem has provided promising results, still, maintaining continued success at detecting malware in operational environments depends on holistically solving challenges regarding the feature variations of malware apps that occur over time and the high costs associated with data labeling. The present study explores the adaptation of the active learning approach for inducing detection models in a non-stationary setting and shows that this approach provides high detection performance with a minimal set of labeled data for a long time when the uncertainty-based sampling strategy is applied. The models that are induced using dynamic, static and hybrid features of mobile malware are compared against baseline approaches. Although active learning has been adapted to many problem domains, it has not been explored in mobile malware detection extensively, especially for non-stationary settings.

Keywords: mobile malware · Android · malware detection · active learning · concept drift · data evolution

1 Introduction

Mobile devices play a significant role in our personal and professional lives. Malicious actors target these devices for various purposes ranging from pursuing economic benefits to collecting information for espionage activities. Mobile malware is one of the greatest cyber threats in this digital ecosystem. Android is the most targeted mobile operating system (OS) by attackers; its share in the threat landscape constitutes 98% of the mobile cyber attacks [10]. The security efforts of Google and device vendors in this regard [4,16] have not been able to put a stop to the increasing trend of this type of cyber attack. Machine learning methods have been proposed to detect malware [20] as these techniques may discriminate behavioral patterns of mobile apps to identify new malicious applications.

S. Goel et al. (Eds.): ICDF2C 2022, LNICST 508, pp. 256–273, 2023.
https://doi.org/10.1007/978-3-031-36574-4_15

The research studies that apply machine learning methods to cyber security problems, in general, and mobile malware detection, in particular, usually validate their results on static data sets belonging to specific time frames (e.g., Drebin [2]). However, the threat landscape is subject to constant evolution due to the inherent attack-defense confrontation between the malicious actors and the security experts in the domain. Relevant dynamic and static features of mobile malware have been proved to continuously change (i.e., legitimate apps are also prone to change to some extent) so that the discrimination capabilities of the learning models diminish over time [9]. Thus, handling the non-stationary property of the data should be one of the building blocks of an operational system to maintain continuous high detection performance for malware detection purposes. This denotes that the learning model should be retrained when a significant data distribution shift is detected. Based on the resources available, one option would be to perform periodic retraining of the model to guarantee an updated model regardless of the variations in the data.

Finding labeled data is always a significant challenge in the cyber security domain due to the lack of human resources or confidentiality concerns that eliminate the possibility of data sharing between different organizations. Although one-class models, which are trained on only legitimate samples, provide a solution to some extent, their performance is usually lower when compared to supervised models and they do not enable the induction of multi-class models which may be highly beneficial to identify malware families. Additionally, they require additional mechanisms to prove that legitimate samples are free from any malicious content, which refers to another form of labeling. Similar to all settings, the effectiveness of non-stationary ones also hugely depends on feeding the training sets with recent samples which are correctly labeled. Therefore, the design considerations of such operational detection systems should holistically address labeling and retraining aspects.

On the other side, despite the aforementioned problems, it does not mean that the cyber security vendors (i.e., companies providing mobile malware scanners or protection products in our case) cannot assign any resources for labeling. A typical vendor has teams of malware analysts that work on a daily basis to investigate the new malware samples. Therefore, we contemplate that both data labeling and model retraining can be achieved by adapting active learning to a non-stationary environment. Active learning approaches create an interactive channel between experts and models so that the models themselves select the most informative samples from an unlabeled data pool, ask the class label from the experts and incorporate their answers into the model. Active learning approaches aim to minimize the labeling efforts to achieve the highest model performance possible. These approaches are very instrumental in cases where obtaining unlabeled data is easy and cheap, but labeling is expensive, which reflects the needs of our target problem [17].

In this study, we adapted a pool-based active learning approach that uses the uncertainty sampling strategy to a non-stationary setting and demonstrated its effectiveness in terms of detection performance and the required labeling effort for mobile malware detection in Android devices. We utilized the Android mal-

ware data set *KronoDroid* [6] which suits well for non-stationary model experiments as it has timestamped malware and *goodware* samples encompassing the whole Android historical timeline. Dynamic and static features of Android apps, more specifically, system calls and permissions, are used for inducing the models separately or in hybrid mode. We compared our results with two baseline models: (1) a *batch retraining* strategy that uses all the previous labeled samples for inducing a model for the next time period, and (2) an iterative learning strategy that randomly selects a sample from the unlabeled data pool without using any informativeness criteria. Our results show that the uncertainty sampling method achieves over 91% F_1 score, on average, throughout a 7-year-long period while enormously minimizing the required labeling effort (i.e., 2–3% of the labeled samples are enough for high detection performance when compared to the batch retraining strategy).

The performance of active learning in non-stationary settings, subject to concept drift issues, has not been demonstrated comprehensively for mobile malware detection. Therefore, this study provides a solid contribution by proposing active learning for addressing the problems of such detection systems in operational settings.

It is important to note that although this study presents experimental results of a solution that covers both data labeling and model retraining, our focus was to elaborate on the trade-off between the labeling effort and its impact on model performance. We assumed that the model is retrained in fixed intervals. It is evident that the detection performance and labeling resource consumption may be also enhanced by different retraining approaches or intervals which can be coupled with various concept drift detectors. However, the detailed analysis of retraining options is out of scope in the present paper.

This paper is structured as follows. Section 2 provides background information and a summary of the related literature. Section 3 provides the methodology while Sect. 4 reports and discusses the main results. Section 5 concludes the study.

2 Background Information and Literature Review

2.1 Background Information

Active Learning. A form of *semi-supervised learning* based on the assumption that a machine learning (ML) algorithm can yield better performance with fewer training iterations (i.e., less data) if it is allowed to select the data from which it learns [18]. For this purpose, a supervised model is trained with a small quantity of data (i.e., *active learner*) and enabled to submit *queries* for selected unlabeled data samples to a labeling *oracle* (i.e., a human expert). The main aim is to achieve high performance using as few labeled samples as possible, thus minimizing the cost of the data labeling process.

The selection of the specific instance for labeling (i.e., query instance) at each training iteration is based on an *informativeness* assessment of the whole set of unlabeled instances performed by the active learner using a specific query

strategy [19]. The pool-based framework, described graphically in Fig. 1, is the most common active learning approach. This approach assumes the existence of a small labeled data set and the availability of a large *pool* of unlabeled data [18]. Query instances are selected from the unlabeled pool for expert annotation. The labeled data sample is then incorporated into the labeled training set which is used to update the knowledge of the ML model (i.e., retraining).

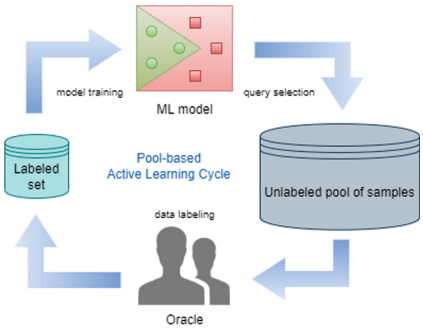

Fig. 1. Pool-based active learning framework

Various query strategies can be used to choose the *most informative* instance [19]. The most commonly used approach is *uncertainty sampling*, where the query instance is selected based on how certain is the learner about the class of the samples. In the *classification uncertainty* scoring strategy, the learner selects the instance (x) for which it is least certain about how to label (i.e., greatest uncertainty). More precisely, the strategy is based on the *least confidence* score (U) computed as:

$$U(x) = 1 - P(y^*|x) \tag{1}$$

where x is a specific instance and y^* is the most likely prediction for that instance.

Concept Drift. Most machine learning models are built on the assumption of *stationary* data, so the testing data attributes are assumed to be similar to the training data attributes, not changing over time. However, in some problem domains, this assumption does not hold as the incoming data features distribution may change over time, thus affecting the *generalization* of the model against new data and, consequently, harming the effectiveness of the detection model over time, a phenomenon called *concept drift* [12]. More precisely, concept drift is observed when the underlying data distribution changes over time (i.e., $P_t(X) \neq P_{t+1}(X)$, where t and $t+1$ indicate consecutive non-overlapping periods) and affects the decision boundary of the classifier which impacts the target class estimation (i.e., $P_t(y|X) \neq P_{t+1}(y|X)$) and results in a decay of performance over time. The appearance of any type of concept drift requires the update of the knowledge of the learning model to the new data distribution to keep high-performance metrics [12].

As a result of the constant evolution of the threat landscape (e.g., new malware) and the inherent evolution of the Android framework (e.g., system updates), Android malware detection systems are prone to concept drift issues. Therefore, an effective malware detection system must update its knowledge to sustain high detection performance over time.

2.2 Literature Review

The first active learning applications in the cyber security domain were carried out for network intrusion detection using the widely-known KDD Cup 1999 data set [1,11]. Uncertainty sampling strategy achieved the reduction of required labeled data by a factor of eight when compared to the baseline strategy, random sampling, [1] whereas confidence measures identified by transductive reliability estimation reduced this factor to the fifth of the same baseline model [11].

Malicious *doc* files were detected by an active learning solution that yielded a high-performance model with 14% of the labeled samples used by the passive learning model [13]. The directory paths that are retrieved from the hierarchical structure of office documents constitute the features of the detection model in the corresponding study. Uncertainty sampling is complemented by a rare-class detection that is applied in the form of multi-class formulation to annotate malware families [3]. The main idea is to include representative samples from all families while selecting the samples from the pool. The proposed approach was applied to two problems, detection of malicious pdf files and network attacks.

A few Android research studies have concentrated on concept drift handling and none of them used the active learning approach in their method. MaMaDroid [14] and DroidEvolver [21] used API calls and traditional ML and online algorithms, respectively, to handle concept drift, whereas [7,8] used system calls and a data stream methodology to tackle the issue. [5] employed the same approach to address concept drift in the permissions feature space.

[15] draws the attention to experimental biases in malware detection research including the temporal bias and demonstrates how validation designs with such biases influence the results obtained. Although it also demonstrates some results regarding the active learning application in non-stationary settings, the purpose is to underline the biases rather than elaborating on an active learning approach. In our study, we investigate the impact of feature types (i.e., dynamic, static, or both) and data balancing strategies on the detection results. We used a data set that encompasses a longer time frame (i.e., our experimentation covers a period of seven years, whereas [15] covers two years of Android data). Our benchmark includes a comparison with random sampling to show the effectiveness of the uncertainty sampling strategy in our problem formulation.

Similarly, active learning is used as one of the methods for maintaining the detection stability of a mobile malware detection model over time in [22]. Although this period encompasses a long period, five years, this study concentrates on the representation of the feature space rather than the comprehensive evaluation of the active learning approach. More specifically, this study proposes

an abstract representation of API call features to grasp better semantic similarities between different malware samples, thus, the model induced using those features can detect the malware evolution better.

3 Methodology

The following sections describe the data set used in our experimental setup, the methodological workflow and the tested scenarios to handle concept drift for Android malware detection using active learning techniques.

3.1 Data Set and Data Features

The data set used in this research is *KronoDroid* [6], a hybrid-featured, labeled, and fully timestamped Android data set, which makes it the most suitable data set for Android concept drift exploration among the available data sets for the purpose of Android malware detection. The *real device* data set was used due to its larger size (i.e., 41,382 malware samples, and 36,755 benign apps). For this study, system calls and permissions features were used as models' input features, along with the *first seen* timestamp and the class labels, which were used to order the samples along the Android historical timeline and class identification, respectively. The *first seen* timestamp, retrieved from *VirusTotal*, provides information about when the sample was received for the first time (i.e., user submission) by the detection system. The usage of this timestamp enabled us to simulate the constant data stream of Android data samples as a realistic scenario for a malware scanner company dealing with an Android malware detection system subject to concept drift issues. For model induction, three sets of input features were used to describe the apps, namely, static (permissions), dynamic (system calls), and hybrid (system calls and permissions) with lengths 166, 288, and 454, respectively, and composed of different variable types. Table 1 summarizes the data set used in this study.

Table 1. Data set summary

Data	Size	Description
Benign samples	36755	Time frame: 2008–2020
Malware samples	41382	Time frame: 2008–2020
Permissions	166	Binary features
System calls	288	Numeric features
Hybrid (perms + syscalls)	454	Binary and numeric features

3.2 Workflow and Scenarios

To explore the application of active learning for concept drift handling and adaptation, the data set was divided into consecutive data chunks, simulating a data stream covering the Android historical timeline. The samples were ordered and grouped in data chunks, according to their timestamp. Additionally, maximum data chunk size and time constraints were used to ensure the existence of sufficient data (i.e., over 100 samples) for every chunk in the whole time frame analyzed. The same classifier algorithm was used in all scenarios and was retrained using different concept drift-handling strategies.

The test scenarios for concept drift handling are described as follows:

- *Batch retraining*: This strategy updates the detection model by retraining the classifier using the whole amount of data available in each specific chunk, and the retrained model is used to forecast the labels for each subsequent period. Therefore, at time t all data from previous time periods (i.e., $s_0, .., s_{t-1}$, where s identifies a data set belonging to a specific time period, t) was used to update the model, and forecast the labels of s_{t+1}. Next, the whole data set belonging to $t+1$ was used to update the model (i.e., retraining) and forecast labels for s_{t+2}. This cycle was repeated for each data chunk until the end of the analysis period. This batch-retraining approach is the frequent solution utilized for concept drift adaptation. It was used as a baseline in our experimentation.
- *Active learning*: This strategy updated the detection model by selecting the *most informative* instances for each data chunk, one at a time, until a predefined performance threshold was reached. The *classification uncertainty* score, as described in Sect. 2.1, was used to rank and select one instance at a time from the unlabeled pool of instances (i.e., whole data chunk). The selected instance was labeled by the *oracle* and used to retrain the model. The rest of the data chunk was used to evaluate the performance increase/decrease after the single retraining step. The training cycle, as depicted in Fig. 1, was repeated until a performance score threshold was achieved. The remaining data, not used in the iterative training steps, were discarded and the trained model was used to forecast all the samples for the next period, as in *batch retraining*. If the performance retrieved processing all the data chunk was lower than the established threshold, the model was rolled back to its best performer configuration and used to forecast the subsequent period data.
- *Random sample selection retraining*: This strategy uses the same iterative training steps as the active learning approach but, in this case, no score is used to select the most informative instances from the unlabeled pool (i.e., whole data chunk). Instead, random sample selection is used. This strategy enabled us to simulate the scenario where a bunch of unlabeled data is available, but no specific criterion is used to select the instances. Thus, samples are selected at random. This model provides the baseline to assess the effectiveness of the sample selection strategy in terms of data labeling minimization.

All the testing scenarios were performed using the same classification algorithm, induced separately using the three feature sets (i.e., static, dynamic, and

hybrid). The performance of the induced models, using the different sets of features for all the strategies, was retrieved and compared. In all cases, the model trained using data from period $t-1$ was used to forecast the labels of the data belonging to the subsequent period, s_t. The main difference among the approaches lies in the training data used and, more specifically, in the strategy used to select the samples for model updating (i.e., all data, random selection or uncertainty score).

The performance of the detection models using the described retraining strategies to handle concept drift was evaluated using two relevant binary classification performance metrics: *accuracy* and F_1 *score* metrics. These metrics were retrieved for each data chunk (i.e., period).

The accuracy metric reports the number of correctly predicted data points out of all the test data points, whereas the F_1 *score* metric is the harmonic mean of *precision* and *recall*. The *precision* of a classification model informs about the fraction of true positive (i.e., malware) data points that the model correctly classified as positive (i.e., malware), while *recall* reports the fraction of samples classified as positive among the total number of positive samples in the testing set.

4 Results and Discussion

The three concept drift-handling retraining strategies described were evaluated using the same base classifier, a Random Forest instance trained using the same initial data set and the default values of Python's *scikit-learn* library. The initial training data set encompassed the months of July and August 2011. This period was selected as it provided enough data to generate a good initial base classification model. Despite that, as the initial training data set was not balanced, a data balancing technique was applied. Two data balancing methods were used (i.e., *random undersampling* and *random oversampling*) and their impact was evaluated. As explained, the remaining data, ordered by their timestamp, were split into consecutive data chunks using temporal and size constraints. Based on experimental tests, the maximum temporal constraint or time window was set at 60 days (i.e., $\approx$2 months) and the maximum data pool size set to 4000. Therefore, the maximum data chunk size was composed of 4000 data samples, spanning a maximum of 60 days of data per chunk. The time period analyzed ranges from the initial time frame (i.e., July-August 2011) to May-June 2018. Posterior time frames did not provide enough quantity of data to continue our experimentation (e.g., chunks with less than 50 samples), so the experimentation time frame and the provided results encompass 7 years of the Android history.

The active learning query strategy was implemented using Python's *modal* library, while the balancing techniques used the *imblearn* library. Given the inherent randomness of some of the strategies (i.e., random selection) and techniques used (i.e., random under/oversampling) each of the evaluated scenarios was repeated 30 times and the average values were reported. The performance threshold to stop processing data for the active learning approaches was set at

0.95 F_1 score. Therefore, if after processing n samples, an F_1 performance of 0.95 (out of 1) or higher was obtained, no more data was labeled in that quarter and the resulting model was used to forecast the labels for the next period data. When the performance threshold was not achieved, the highest F_1 performer model was used.

Table 2 provides the obtained results using all the described concept drift-handling approaches. More specifically, the column *feature set* describes the input features used by each specific model tested and the *balancing method* column reports the technique used to balance the initial data set, in the case of the two query strategies used (i.e., random and uncertainty), as well as all data chunks for the batch approach (i.e., to avoid that the imbalanced data chunks generated biased RF models). For each combination of the feature sets and balancing methods, three strategies to handle concept drift were evaluated, described in Sect. 3.2, and referenced in the *query strategy* column. The remaining columns in Table 2 report the performance metrics that enabled us to analyze and compare all the evaluated approaches. The *labeled samples* column informs about the average number of samples processed by each model (i.e., $\overline{x}$), that is, the number of instances labeled, to reach the performance threshold, $F_1 \geq 95\%$. The columns F_1 *score* and *accuracy* provide the average performance of the trained models in all time windows in the analyzed time frame (e.g., 45 data chunks spanning between September/October 2011 and May/June 2018). The reported values for labeled samples and the performance metrics are the average values of the 30 tests performed for each specific scenario. The standard deviation (i.e., s) is reported to contextualize better the mean value as a data descriptor. Additionally, for the *labeled samples*, the proportion of the average number of labeled samples reported in relation to the total data available in the analyzed period is reflected by the % column.

As can be observed in Table 2, when the permissions features are used, the active learning approach (i.e., uncertainty) provides similar performance as the baseline model (i.e., batch, using all data), but requires the smallest number of data samples among the tested strategies. The uncertainty-based active learning approach minimizes the data labeling needed to achieve similar performance as the other two approaches using either of the balancing techniques. More precisely, the batch approach, which requires the labeling of all the data samples, shows slightly better performance than the active learning approaches, but these show significantly lower data labeling requirements. In this regard, the uncertainty-based active learning approach outperforms the random selection approach by using less than 18% of the total data in both cases. Even though both single query-based retraining approaches show benefits over the batch approach, the active learning approach requires three times fewer data than the random instance selection to achieve the same performance metrics. This fact evidences that, in the permissions case, the single query-based gradual modification of the classifier decision boundary shows benefits when it is compared to the baseline model, which uses batch processing (all data). The random approach shows slightly lower performance than the baseline model, but with less data labeling

Table 2. Testing scenarios results

Feature set	Balancing method	Query strategy	Labeled samples			F_1 score		Accuracy	
			$\overline{x}$	s	%	$\overline{x}$	s	$\overline{x}$	s
Permissions	Oversampling	Batch	67068	0	100	91.2	0.4	92.5	0.4
		Random	30100.4	129.8	44.9	89.4	1.0	90.3	1.0
		Uncertainty	11845.6	41.7	17.7	89.4	0.6	90.4	0.6
	Undersampling	Batch	67068	0	100	90.9	0.8	92.4	0.8
		Random	29409.9	113.5	43.9	89.5	1.1	90.3	1.1
		Uncertainty	9281.4	35.5	13.8	89.6	0.7	90.5	0.6
Syscalls	Oversampling	Batch	67068	0	100	85.1	0.8	86.1	0.7
		Random	45028.9	127.8	67.1	84.1	0.9	84.9	0.9
		Uncertainty	13098.8	38.6	19.5	84.5	0.9	85.3	0.8
	Undersampling	Batch	67068	0	100	82.7	1.2	83.3	1.2
		Random	45378.7	118.5	67.7	84.5	0.9	85.0	1.0
		Uncertainty	12748.3	52.3	19.0	85.1	1.0	85.5	1.0
Hybrid	Oversampling	Batch	67068	0	100	92.8	0.5	93.5	0.4
		Random	22057.2	121.5	32.9	90.9	1.0	91.2	1.0
		Uncertainty	1991.9	8.9	3.0	91.6	1.2	91.9	1.2
	Undersampling	Batch	67068	0	100	92.5	0.6	93.1	0.6
		Random	20978.4	116.1	31.3	91.0	1.1	91.1	1.1
		Uncertainty	1459.4	6.3	2.2	91.7	1.4	91.9	1.4

needs, evidencing that more data might not be necessary to handle concept drift effectively but that, more importantly, the instance-based gradual retraining of the model may be more beneficial to handle concept drift effectively. There are no major differences in performance in any cases when both balancing methods are compared. However, the *undersampling* approach provides similar performance metrics to the *oversampling* method with significantly fewer data in the *active learning* case (i.e., 28% more data is needed, on average, for the oversampling case than for the undersampling scenario). In conclusion, the best results in terms of both performance and minimization of data labeling needs, when the permissions feature set is used, are obtained using the undersampling balancing strategy combined with the uncertainty-based active learning query approach.

Comparatively, the system calls feature set produced the worst performance models among all tested models in both evaluated metrics, the number of labeled samples needed and performance achieved. More precisely, the batch strategy using undersampling provides average performance metrics below 85%, which are slightly better when oversampling is used. These performance metrics are the worst across all models and feature sets, as none of the tests using the permissions or the hybrid feature sets go lower than 89.4% F_1 and 90.3% accuracy. However, the system calls-based model performance is significantly improved when a single query strategy is implemented, and, more specifically, when the uncertainty-based active learning approach is used, reaching similar performance

as the baseline model, as in the permissions case, and even outperforming when undersampling is used. Despite that, the labeling needs for the uncertainty-based active learning, which, again, minimizes the labeling cost, is superior to the permissions case for both sampling techniques (i.e., a minimum of 19% of the data has to be labeled by the oracle). It is worth noting that random selection reaches similar performance as the uncertainty-based strategy but requires, in both cases, over 66.7% of the whole data to be labeled. Thus, again, the single query approach shows advantages over batch processing to handle concept drift effectively, especially when the uncertainty-based active learning approach is applied.

The hybrid feature sets, which combine the permissions and system calls sets for model induction, provide the best overall models, in all cases. The active learning approach using the uncertainty criterion reaches a slightly lower performance than the baseline performance implementing the batch approach. However, in this case, the benefits of the active learning approach are especially evident for both balancing techniques. The labeling needs are significantly lowered, not over-passing 9% of the whole data set. As a result, they provide the best performance-labeling trade-off results among all the test scenarios. In this regard, the best model of all the tested scenarios is obtained using the active learning approach combined with undersampling, yielding an average 91.7% F_1 score and 91.6% accuracy using, on average, only 1460 samples (i.e., ≈2.2% of the total data) to provide effective detection in the seven-year-long study period. Comparatively, the uncertainty-based active learning approach for the hybrid-featured models requires 10–15 times fewer data than the random query approach to achieve better performance results and 50 times fewer data to reach similar detection performance than the baseline models. These results show that the hybrid feature set generates better discriminatory models which benefit notably from the active learning approach, being able to handle concept drift with a significantly reduced quantity of labeled data belonging to specific time frames along a seven-year-long time period (i.e., from September-October 2011 to May-June 2018).

To further explore the results, the summary values reported in Table 2 are provided in more fine-grained detail on the analyzed historical timeline of Android in Fig. 2, 3, 4, and 5. More specifically, in these figures, the X-axis reports the time frame of the specific data chunk, encompassing, at maximum, two months of data. The axis labels provide the year and month separated by a slash (e.g., 2011/9–10 reports data comprised between September and October 2011). The left Y-axis reports the number of samples included in every data chunk (i.e., grey color), thus composing the unlabeled pool of samples for the active learning approaches, that were actually labeled by the oracle (i.e., blue color). Given the degree of randomness of the approaches used, the reported values for the number of labeled samples (i.e., blue area on the bars) are mean values with the confidence interval of the mean estimation reported by the white whiskers that extend over and below the mean (i.e., confidence level 95%). The average performance scores obtained on each data chunk are reported by the

yellow (i.e., accuracy) and blue (i.e., F_1 score) lines placed on top of the bar chart, and ranging from 0 to 1 (i.e., right Y-axis). The standard deviation of these performance metrics is provided by the colored ribbons surrounding the average lines. Figure 2 provides the average results for the uncertainty-based active learning approach when undersampling and the permissions set were used. Figure 3 reports the same information when the system calls set is used while Fig. 4 provides the hybrid feature set-related information. These figures enable us to compare the impact of the feature set under the same conditions (i.e., uncertainty-based active learning approach using undersampling). Lastly, Fig. 5 enables the comparison between the best active learning model (i.e., hybrid feature set, undersampling using uncertainty score, as depicted in Fig. 4) and the random query strategy for the same feature set and sampling approach configuration.

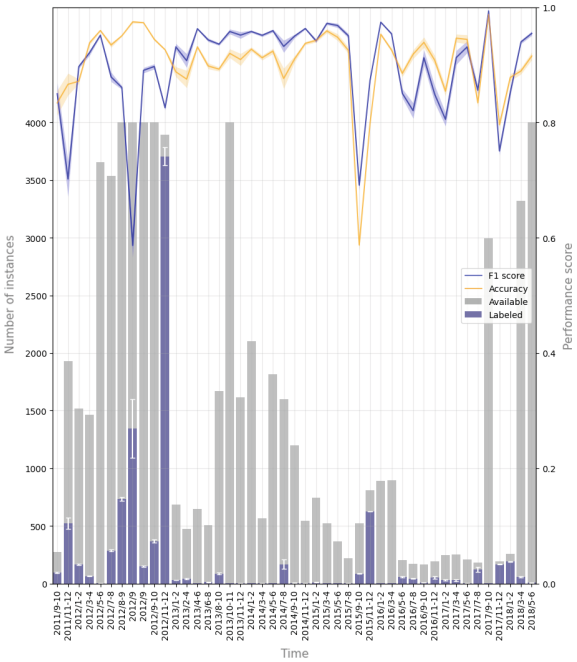

Fig. 2. Permissions, undersampling, and uncertainty-based model results

As can be observed, in Fig. 2, the permissions feature set enabled the handling of concept drift using significantly less labeled data than the system calls feature set, depicted in Fig. 3. With some minor exceptions (e.g., 11–12/2012), the permissions feature set required fewer labeled data per chunk to sustain the training target of 95% F_1 score, over-passing this score in many chunks, thus no data was labeled for training purposes (e.g., 10–11/2013, 11–12/2013, 1–2/2014,

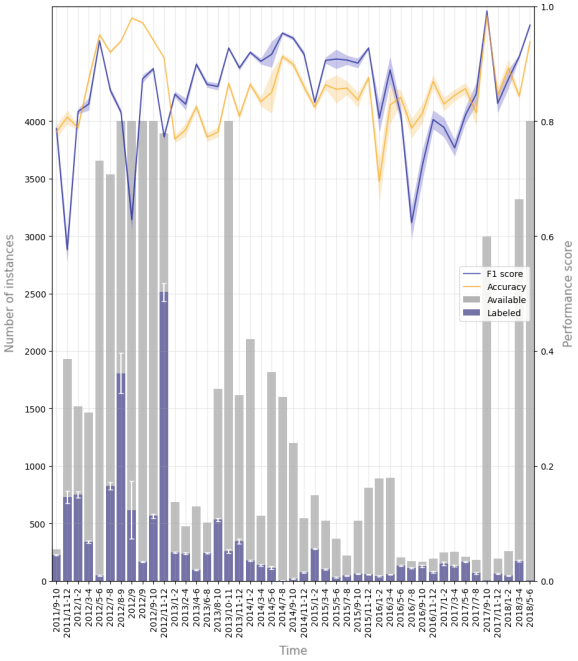

Fig. 3. System calls, undersampling, and uncertainty-based model results

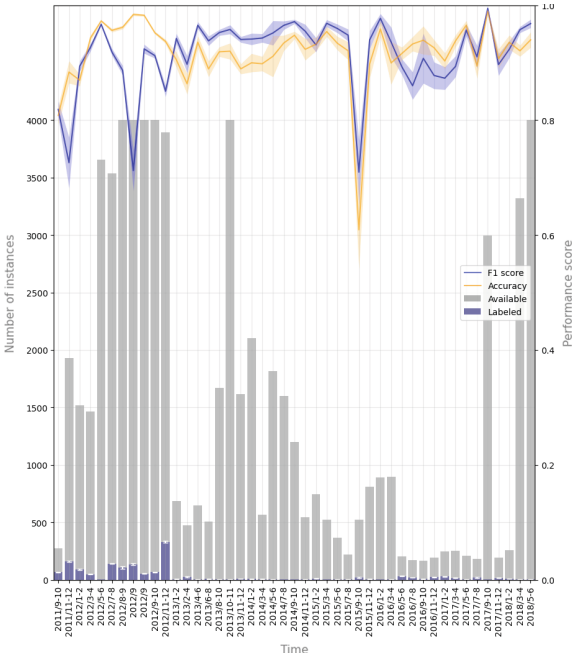

Fig. 4. Hybrid, undersampling, and uncertainty-based model results

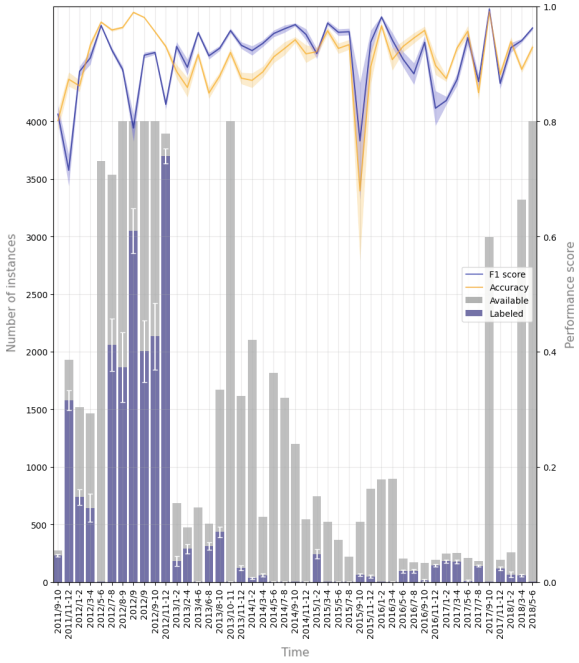

Fig. 5. Hybrid, undersampling, and random selection model results

3–4/2014, and 5–6/2014). Despite the goodness shown by the permissions feature set to handle concept drift using the active learning approach, these results are significantly outperformed by the hybrid feature set, which combines the system calls and permissions feature set. In this case, a reduced proportion of the chunk data is labeled in every chunk to achieve high-performance metrics (e.g., 9–10/2011, 11–12/2011) with extended periods of almost no training data needs (e.g., from 4–6/2013 to 7–8/2015). Therefore, the high-dimensional feature space generated by the joint usage of both feature sets enabled the handling of concept drifts better than any other approach, keeping high-performance metrics with just a few samples labeled per chunk. Even though this feature set reduces the data needs in all approaches and strategies, the uncertainty-based query strategy shows significant improvement concerning random query selection, as can be seen in Fig. 5. The random query strategy requires significantly more labeled data per data chunk to sustain performance and address concept drift, evidencing the superiority of the uncertainty-based selection over random query selection.

The obtained results show that the active learning approach, in its most basic form (i.e., uncertainty sampling) can be effectively used to handle concept drift, keeping high-performance metrics while minimizing the data labeling efforts (i.e., the amount of labeled data needed to keep high performance). As a result, active learning might be an efficient and effective solution to handle

concept drift in environments where a large quantity of unlabeled data is available but with high labeling costs. It allows focusing the labeling effort on the *relevant* data to improve the model and discard the *irrelevant* data samples that may not provide benefit to the model. Despite that, uncertainty sampling may yield *biased* classifiers and sub-optimal models if the initial data set is too small or not representative as the model *certainty* is used to rank the *informativeness* or relevancy of the samples. To avoid that, other query strategies could be used such as query by committee or ranked batch-mode [18]. In our case, the initial data set has been proved reliable and large enough to overcome this issue and the performance obtained by the models does not change significantly (i.e., standard deviation values are not large). The exploration of the benefits of other approaches constitutes part of our future work.

The comparative performance metrics provided in this study show that the gradual modification of the decision boundary caused by the addition of a single relevant sample in the training data set provides high-performance models using significantly less data than the batch retraining approach. Random instance selection improves the labeling needs concerning the batch retraining approach, but they are both outperformed significantly by the active learning approach. Random selection requires consistently more data to achieve roughly the same (but not better) performance metrics than the uncertainty sampling approach. This fact evidences the goodness of the active learning approach to induce great performance models with significantly fewer data needs.

To address imbalance issues, two balancing techniques were explored. Random oversampling balances the data by generating artificial but similar data points for the underrepresented class, whereas random oversampling selects a random sample from the overrepresented class to match the number of samples in the underrepresented class. Even though both approaches worked similarly for random and batch strategies, the undersampling approach provided distinctive benefits using the active learning approach for the permissions and hybrid feature sets. This technique minimized the data labeling efforts significantly while producing great discriminatory results. The exploration of more complex balancing approaches is part of our future work.

This paper explores the application of *active learning* as an alternative approach to deal with concept drift in Android malware detection. The related methods in the literature [14,15,21,22] propose the usage of more complex algorithmic solutions that require extensive data labeling efforts and intensive computational resources. The active learning approach, due to its focus on data labeling minimization, reduces the computational load and resources needed to deal effectively with concept drift issues, as demonstrated in this paper. More specifically, the comparison of the results obtained in this study with related works [5,7,14,15,21,22] evidences the goodness of the active learning approach to maximize detection performance metrics while minimizing labeling needs and, consequently, computational resources. Most of these proposed solutions assume the labeling of the whole data set at each training step thus they are analogous to the batch retraining approach, which was used as a baseline in our study. Besides,

the detection solutions in the literature are more computationally intensive due to their algorithmic complexity. For instance, some methods combine the output of a pool of classifiers [7,21] or use complex adaptive pipelines [14] that increase the burden of system maintenance and the overall resources needed to operate and update the detection system. In our benchmarking, a single classifier model, based on a traditional machine learning algorithm (i.e., Random Forest), using the *active learning* query strategy was capable of providing high detection performance for a long period (i.e., from 2011 to 2018) with few data updates over time and many time frames with zero labeling needs. The performance results are similar to the ones proposed by [7] and the baseline approach (i.e., batch retraining), but use a less complex system, which is easier to maintain and requires less computational and labeling resources. It also outperforms the rest of the concept drift-handling methods in the related literature, which manifest significant performance decay over time [14,21].

5 Conclusions

The active learning approach is built on the assumption that a machine learning model can learn faster (i.e., in fewer training steps) and with less data if the model is allowed to select the data from which it learns. This approach combines the knowledge of an *oracle* and instance selection by the supervised model to enhance performance and minimize data needs. To the best of our knowledge, this is the first study that leverages *active learning* to handle concept drift in Android malware detection. Our results show that the active learning approach, in its most basic form, allows effective concept drift handling in Android malware detection and, more interestingly, minimizes the data labeling needs. Consequently, it becomes an option worth considering for enhancing the ML-based detection systems in cyber security environments (e.g., malware protection companies, security operations centres dealing with Android malware detection), where a large body of unlabeled data is constantly available but the high labeling cost associated makes the task infeasible and prohibitive, thus affecting the detection capabilities of the system.

Acknowledgments. This work is partially funded by the European Union's Horizon 2020 Research and Innovation Programme through ECHO (https://echonetwork.eu/) project under Grant Agreement No. 830943.

References

1. Almgren, M., Jonsson, E.: Using active learning in intrusion detection. In: 2004 Proceedings of the 17th IEEE Computer Security Foundations Workshop, pp. 88–98. IEEE (2004)
2. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C.: Drebin: effective and explainable detection of android malware in your pocket. In: NDSS, vol. 14, pp. 23–26 (2014)

3. Beaugnon, A., Chifflier, P., Bach, F.: ILAB: an interactive labelling strategy for intrusion detection. In: Dacier, M., Bailey, M., Polychronakis, M., Antonakakis, M. (eds.) RAID 2017. LNCS, vol. 10453, pp. 120–140. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66332-6_6
4. Google: Google play protect (2021). https://developers.google.com/android/play-protect
5. Guerra-Manzanares, A., Bahsi, H., Luckner, M.: Leveraging the first line of defense: a study on the evolution and usage of android security permissions for enhanced android malware detection. J. Comput. Virol. Hacking Tech. **19**, 1–32 (2022)
6. Guerra-Manzanares, A., Bahsi, H., Nõmm, S.: KronoDroid: time-based hybrid-featured dataset for effective android malware detection and characterization. Comput. Secur. **110**, 102399 (2021)
7. Guerra-Manzanares, A., Luckner, M., Bahsi, H.: Android malware concept drift using system calls: detection, characterization and challenges. Expert Syst. Appl. 117200 (2022). https://doi.org/10.1016/j.eswa.2022.117200
8. Guerra-Manzanares, A., Luckner, M., Bahsi, H.: Concept drift and cross-device behavior: challenges and implications for effective android malware detection. Comput. Secur. **120**, 102757 (2022). https://doi.org/10.1016/j.cose.2022.102757
9. Guerra-Manzanares, A., Nomm, S., Bahsi, H.: In-depth feature selection and ranking for automated detection of mobile malware. In: ICISSP, pp. 274–283 (2019)
10. Kaspersky: Mobile security: Android vs ios - which one is safer? (2020). https://www.kaspersky.com/resource-center/threats/android-vs-iphone-mobile-security
11. Li, Y., Guo, L.: An active learning based TCM-KNN algorithm for supervised network intrusion detection. Comput. Secur. **26**(7–8), 459–467 (2007)
12. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2018)
13. Nissim, N., Cohen, A., Elovici, Y.: ALDOCX: detection of unknown malicious Microsoft office documents using designated active learning methods based on new structural feature extraction methodology. IEEE Trans. Inf. Forensics Secur. **12**(3), 631–646 (2016)
14. Onwuzurike, L., Mariconti, E., Andriotis, P., Cristofaro, E.D., Ross, G., Stringhini, G.: MaMaDroid: detecting android malware by building Markov chains of behavioral models (extended version). ACM Trans. Priv. Secur. **22**(2) (2019). https://doi.org/10.1145/3313391
15. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: {TESSERACT}: eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 2019), pp. 729–746 (2019)
16. Samsung: About knox (2021). https://www.samsungknox.com/en/about-knox
17. Schütze, H., Velipasaoglu, E., Pedersen, J.O.: Performance thresholding in practical text classification. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 662–671 (2006)
18. Settles, B.: Active learning literature survey (2009)
19. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp. 1070–1079 (2008)

20. Sharma, T., Rattan, D.: Malicious application detection in android - a systematic literature review. Comput. Sci. Rev. **40**, 100373 (2021)
21. Xu, K., Li, Y., Deng, R., Chen, K., Xu, J.: DroidEvolver: self-evolving android malware detection system. In: 2019 IEEE European Symposium on Security and Privacy (EuroS P), pp. 47–62 (2019). https://doi.org/10.1109/EuroSP.2019.00014
22. Zhang, X., et al.: Enhancing state-of-the-art classifiers with API semantics to detect evolved android malware. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 757–770 (2020)